The SEMAINE API:
An open-source research platform for multimodal, emotion-oriented interactive systems

Marc Schröder, DFKI
eINTERFACE 2010, Amsterdam
Outline

- The SEMAINE project
- The SEMAINE API
  - Motivation
  - A component integration framework
  - API architecture
  - Markup interfaces
- Building new emotion-oriented systems with the SEMAINE API
  - combining existing and new modules
- Overview of Tutorial
The SEMAINE project

“Sustained Emotionally coloured Machine-human Interaction using Nonverbal Expression”
- http://www.semaine-project.eu

FP7 project, Jan. 2008 – Dec. 2010

Aim: build an autonomous, responsive Sensitive Artificial Listener (SAL) system
- real-time multimodal interaction
- strong on non-verbal skills
- cut down on verbal skills for now
- conversation: chat, not task-oriented
The SEMAINE project: Partners

- Coordinator: DFKI Saarbrücken (M. Schröder)
  - system integration, speech synthesis
- Queen's University Belfast (R. Cowie)
  - data collection, evaluation
- Imperial College, London (M. Pantic)
  - facial analysis
- University of Twente (D. Heylen)
  - dialogue modelling
- CNRS (C. Pelachaud)
  - ECA system
- TU München (B. Schuller)
  - ASR, vocal emotion recognition
The SEMAINE system
The SEMAINE system
The SEMAINE system: Components

- Facial analysis (Windows / Mac, C++)
  - face detector, nod/shake detector
- Voice analysis (Linux / Mac, C++)
  - feature extraction, keyword spotting, emotion rec.
- Verbal utterance planning (Java)
  - interpret user behaviour, plan agent behaviour
- Listener behaviour (Windows, C++)
  - generate backchannels + feedback
- Speech synthesis (Java)
- Visual behaviour realisation (Windows, C++)
Emotion-oriented interactive systems

- Emotion-enabled interactive characters
  - MAX
  - FearNot: learning anti-bullying strategies
  - IDEAS4Games
  - ...

- Other emotion-related functionality
  - AffectiveDiary
  - eMoto: Pen for affective text messages
  - ...

Marc Schröder, DFKI
Research systems tend to create ad hoc systems with domain-specific representations

Difficult to reuse

Difficult to combine parts of existing systems to implement new functionality
The benefit of standards

- Standards are taken for granted in many parts of our life
  - electric voltage (220 V throughout Europe)
  - fuel grade (95 Octane)
  - screw threads
  - ...
  - more recently: Document formats (ODF)

- Users can rely on guaranteed properties without worrying who provides the service and how it is implemented
  - supports interoperability and reuse
The SEMAINE API

- Integrate components across languages and operating systems: Middleware!
- Message-oriented middleware (MOM)
  - asynchronous
    - sender doesn't need to wait for receiver
  - publish-subscribe to "Topics"
    - flexible n-to-m message passing
  - currently using Apache ActiveMQ, a JMS server
    - open source, Java/C++, reasonably fast
- SEMAINE API is a component integration layer on top of the MOM
“Reasonably fast”: ActiveMQ vs. Psyclone

- Round-trip message routing times on localhost
  -_activeMQ_
    - 0.3 – 55 ms
  - Psyclone
    - 16 – 408 ms
  - ActiveMQ between 10 and 50 times faster than Psyclone
The SEMAINE API: System architecture

message-oriented middleware

semaine.data.*

semaine.callback.*

semaine.log.*

semaine.meta

log reader

system manager

system monitor

GUI
The SEMAINE API: System monitor

System status:
- System status: ready
time: 36173
average...
...time spent in act() 1 ms
...time spent in react0 162 ms
...message travel time (ignoring clock async): 894879 ms
...number of messages received: 999

Log messages:
- all components

system to become ready
INFO semaire.log.PlayerOgre 2009-10-16 12:33:52.539 waiting for system to become ready
INFO semaire.log.MessageLogComponent 2009-10-16 12:33:46.462 waiting for system to become ready
INFO semaire.log.UserPresenceInterpreter 2009-10-16 12:33:46.463 waiting for system to become ready
INFO semaire.log.LiteranceActionProposer 2009-10-16 12:33:46.464 waiting for system to become ready
INFO semaire.log.HumanIntentPlanner 2009-10-16 12:33:52.545 waiting for system to become ready
INFO semaire.log.HeadMovementInterpreter 2009-10-16 12:33:46.498 waiting for system to become ready
INFO semaire.log.LiteranceRealiser 2009-10-16 12:33:46.624 waiting for system to become ready
INFO semaire.log.SystemManager 2009-10-16 12:33:47.431 System is ready
INFO semaire.log.ListenerIntentPlanner 2009-10-16 12:33:53.516 system ready
- let's go
INFO semaire.log.HumanIntentRealiser 2009-10-16 12:33:53.517 system ready - let's go
INFO semaire.log.VideoFeatureExtractor 2009-10-16 12:33:53.517 system ready
- let's go
INFO semaire.log.PlayerOgre 2009-10-16 12:33:53.517 system ready - let's go
INFO semaire.log.HumanIntentPlanner 2009-10-16 12:33:53.517 system ready - let's go
INFO semaire.log.HeadMovementInterpreter 2009-10-16 12:33:47.440 system ready - let's go
INFO semaire.log.TurnFeatureExtractor 2009-10-16 12:33:47.440 system ready - let's go
INFO semaire.log.MessageLogComponent 2009-10-16 12:33:47.443 system ready - let's go
INFO semaire.log.SpeechPreprocessor 2009-10-16 12:33:47.448 system ready - let's go
INFO semaire.log.SpeechMLRealiser 2009-10-16 12:33:47.452 system ready - let's go
INFO semaire.log.UserPresenceInterpreter 2009-10-16 12:33:47.458 system ready - let's go
INFO semaire.log.LiteranceRealiser 2009-10-16 12:33:47.463 system ready - let's go
INFO semaire.log.EmotionInterpreter 2009-10-16 12:33:47.465 system ready - let's go
INFO semaire.log.TurnTakingInterpreter 2009-10-16 12:33:47.466 system ready - let's go
INFO semaire.log.PersonalControl 2009-10-16 12:33:47.467 system ready - let's go
INFO semaire.log.SpeechInterpreter 2009-10-16 12:33:47.468 system ready - let's go
INFO semaire.log.GUI 2009-10-16 12:33:47.470 system ready - let's go

Marc Schröder, DFKI
Representation formats in the SEMAINE API

EMMA w/ EmotionML or BML

- Interpreters
  - user state
  - dialog state
  - agent state
  - Action proposers

- Interpreters
- FML

- Action selection
  - action
  - Behaviour planner
    - behaviour plan
    - Behaviour realiser
      - behaviour data
      - Player

- SemaineML

- Analysers
  - features
  - feature vectors

- Feature extractors

Marc Schröder, DFKI
EMMA: W3C Extensible Multimodal Annotation markup language

- container format for user behaviour analysis

EMMA transporting EmotionML:

```xml
<emma:emma xmlns:emma="http://www.w3.org/2003/04/emma" version="1.0">
  <emma:interpretation emma:start="123456789">
    <emotion xmlns="http://www.w3.org/2005/Incubator/emotion">
      <dimensions set="valenceArousalPotency">
        <arousal value="-0.29"/>
        <valence value="-0.22"/>
      </dimensions>
    </emotion>
  </emma:interpretation>
</emma:emma>
```
EmotionML

- W3C working draft

Representation of emotions informed by affective sciences

- categories, dimensions, appraisals, action tendencies
- links to triggers, objects, expressive behaviour
- customizable: choose vocabulary to use
EMMA: W3C Extensible Multimodal Annotation markup language
- container format for user behaviour analysis

EMMA transporting BML:
<emma:emma xmlns:emma="http://www.w3.org/2003/04/emma" version="1.0">
  <emma:interpretation emma:start="123456789">
  </emma:interpretation>
</emma:emma>
SemaineML

- custom format for domain-specific annotations

SemaineML representing dialogue state:
<dialog-state xmlns="http://www.semaine-project.eu/semaineml" version="0.0.1">  
  <speaker who="agent"/>  
  <listener who="user"/>  
</dialog-state>
BML: Behaviour Markup Language

- drive ECA behaviour
- speech, face, and gesture

- first, symbolic representation of AV synchronization
- TTS adds detailed timings
BML with symbolic time markers:

```xml
<bml xmlns="http://www.mindmakers.org/projects/BML" id="bml1">
  <speech id="s1" language="en-US" text="Hi, I'm Poppy." ssml:xmlns="http://www.w3.org/2001/10/synthesis">
    <ssml:mark name="s1:tm1"/>
    Hi,
    <ssml:mark name="s1:tm2"/>
    I'm
    <ssml:mark name="s1:tm3"/>
    Poppy.
    <ssml:mark name="s1:tm4"/>
    <pitchaccent id="xpa1" start="s1:tm1" end="s1:tm2"/>
    <pitchaccent id="xpa2" start="s1:tm3" end="s1:tm4"/>
    <boundary id="b1" time="s1:tm4"/>
  </speech>
  <gaze id="g1" start="s1:tm1" end="s1:tm4">
    ...
  </gaze>
  <head id="h1" start="s1:tm3" end="s1:tm4" type="NOD">
    ...
  </head>
</bml>
```
BML with symbolic time markers:

```xml
<bml xmlns="http://www.mindmakers.org/projects/BML" id="bml1">
  <speech id="s1" language="en-US" text="Hi, I'm Poppy." ssml:xmlns="http://www.w3.org/2001/10/synthesis">
    <ssml:mark name="s1:tm1"/>
    Hi,
    <ssml:mark name="s1:tm2"/>
    I'm
    <ssml:mark name="s1:tm3"/>
    Poppy.
    <ssml:mark name="s1:tm4"/>
    <pitchaccent id="xpa1" start="s1:tm1" end="s1:tm2"/>
    <pitchaccent id="xpa2" start="s1:tm3" end="s1:tm4"/>
    <boundary id="b1" time="s1:tm4"/>
  </speech>
  <gaze id="g1" start="s1:tm1" end="s1:tm4">
    ...
  </gaze>
  <head id="h1" start="s1:tm3" end="s1:tm4" type="NOD">
    ...
  </head>
</bml>
```

Marc Schröder, DFKI
BML with symbolic time markers:

```xml
<bml xmlns="http://www.mindmakers.org/projects/BML" id="bml1">
  <speech id="s1" language="en-US" text="Hi, I'm Poppy." ssml:xmlns="http://www.w3.org/2001/10/synthesis">
    <ssml:mark name="s1:tm1"/>
    Hi,
    <ssml:mark name="s1:tm2"/>
    I'm
    <ssml:mark name="s1:tm3"/>
    Poppy.
    <ssml:mark name="s1:tm4"/>
    <pitchaccent id="xpa1" start="s1:tm1" end="s1:tm2"/>
    <pitchaccent id="xpa2" start="s1:tm3" end="s1:tm4"/>
    <boundary id="b1" time="s1:tm4"/>
  </speech>
  <gaze id="g1" start="s1:tm1" end="s1:tm4">
    ...
  </gaze>
  <head id="h1" start="s1:tm3" end="s1:tm4" type="NOD">
    ...
  </head>
</bml>
```

Marc Schröder, DFKI
BML with phone timing for lip synchronization:

```xml
<bml xmlns="http://www.mindmakers.org/projects/BML" id="bml1">
    <speech id="s1" language="en_US" text="Hi, I'm Poppy.
        ssml:xmlns="http://www.w3.org/2001/10/synthesis"
        mary:xmlns="http://mary.dfki.de/2002/MaryXML">
        ...
        <ssml:mark name="s1:tm3"/>
        Poppy.
        <mary:syllable stress="1">
            <mary:ph d="0.092" end="1.011" p="p"/>
            <mary:ph d="0.112" end="1.123" p="A"/>
            <mary:ph d="0.093" end="1.216" p="p"/>
        </mary:syllable>
        <mary:syllable>
            <mary:ph d="0.141" end="1.357" p="i"/>
        </mary:syllable>
        <ssml:mark name="s1:tm4"/>
        ...
    </speech>
</bml>
```
Some representations available from standards or standards-in-the-making

- EMMA, EmotionML, BML, SSML

Others still missing

- FML, dialogue state, …

Implementation provides “reality check” for a specification

- feedback from implementation can improve spec
Building new emotion-oriented systems

- SEMAINE API simplifies the integration task
  - abstract away from operating system, programming language and communication issues
  - use standard representation formats where possible
  - provide suitable support for XML handling

- Three example systems:
  - Hello world
  - Emotion mirror
  - The swimmer's game
1. Emotional Hello world

- Minimal system:
  - text input component
  - dummy text analysis to infer emotional state
  - emoticon output

- Code is short – about 20 lines each
1. Emotional Hello world

Valence
- 0 +

Arousal
+ 8- ( 8- | 8-)
0 :-( :: ::)
- *- ( *- *-)

EmotionML

Marc Schröder, DFKI
public class HelloAnalyser extends Component {

private XMLSender emotionSender =
        new XMLSender("semaine.data.hello.emotion", "EmotionML", getName());

public HelloAnalyser() throws JMSException {
    super("HelloAnalyser");
    receivers.add(new Receiver("semaine.data.hello.text");
    senders.add(emotionSender);
}

@Override protected void react(SEMAINEMessage m) throws JMSException {
    int arousalValue = 0, valenceValue = 0;
    String input = m.getText();
    if (input.contains("very")) arousalValue = 1;
    else if (input.contains("a bit")) arousalValue = -1;
    if (input.contains("happy")) valenceValue = 1;
    else if (input.contains("sad")) valenceValue = -1;
    Document emotionML = createEmotionML(arousalValue, valenceValue);
    emotionSender.sendXML(emotionML, meta.getTime());
}
2. Emotion mirror

- Mimick user emotion
  - Infer user emotion from user speech
  - Represent inferred emotion as an emoticon
- Benefit of reuse: need to implement only one new component
  - Reuse openSMILE emotion recognition from SEMAINE system
  - Reuse EmoticonOutput from Hello World example
  - Implement simple “decision” component that extracts emotion judgements from recognition output
2. Emotion mirror
2. Emotion mirror

```java
public class EmotionExtractor extends Component {
    private XMLSender emotionSender =
        new XMLSender("semaine.data.hello.emotion", "EmotionML", getName());

    public EmotionExtractor() throws JMSException {
        super("EmotionExtractor");
        receivers.add(new EmmaReceiver("semaine.data.state.user.emma"));
        senders.add(emotionSender);
    }

    @Override
    protected void react(SEMAINEMessage m) throws JMSException {
        SEMAINEEmmaMessage emmaMessage = (SEMAINEEmmaMessage) m;
        Element interpretation = emmaMessage.getTopLevelInterpretation();
        List<Element> emotionElements = emmaMessage.getEmotionElements(interpretation);
        if (emotionElements.size() > 0) {
            Element emotion = emotionElements.get(0);
            Document emotionML = XMLTool.newDocument(EmotionML.ROOT_ELEMENT,
                                                    EmotionML.namespaceURI);
            emotionML.adoptNode(emotion);
            emotionML.getDocumentElement().appendChild(emotion);
            emotionSender.sendXML(emotionML, meta.getTime());
        }
    }
}
```
3. The Swimmer's Game

- Simple emotional speech-driven game
  - a swimmer needs to reach the river bank, but is pulled by the water towards the waterfall
  - user can cheer up the swimmer with aroused speech

- Components:
  - emotion detection from speech
  - position computer for swimmer
  - GUI display of swimmer position
  - commentator using TTS output
3. The Swimmer's Game
3. The Swimmer's Game
SEMAINE API: Summary

- SEMAINE API makes it easy to write new components in Java / C++
- Integration simplified by standard representation formats
What you will learn in the Tutorial

- Install the SEMAINE 2.0 system
- Understand parts involved:
  - MOM ActiveMQ
  - Java components
  - native components
  - distributed system
- Write a SEMAINE component in Java/C++
- Create a new emotion-oriented system from new and existing components